15-451 Algorithms, Spring 2017
Recitation #10 Worksheet

Counting Axis-Aligned Segment Intersections

You're given n vertical line segments in the plane and m horizontal ones. The problem is to count the number of intersections. The brute-force algorithm is to try intersecting all n verticals with all m horizontals, which is $O(nm)$. Devise a solution based on SegTrees that is $O((n+m)\log(n+m))$.

Hint 1: Take advantage of the fact that there are only $2n+m$ relevant y coordinates where anything interesting happens.

Hint 2: Sweep from left to right, processing each event as it comes. The events are: (1) a vertical segment appears, (2) a left end of a horizontal segment appears and (3) the right end of a horizontal segment appears.

Solution: Following Hint 1, collect all the y-coordinates where something happens, and then use a hash table to map each of these y coordinates to the integers $1, \ldots N$, such that the map preserves sorted order. Here $N \leq 2n+m$ is the number of y coordinates where something happens.

Now set up a SegTree to represent N variables. These are initially zero. When we process a left endpoint of a segment we increment the appropriate variable for that y coordinate. Right endpoints decrement the appropriate variable. Finally, for a vertical segment, we do a range sum for the values of the variables between the two y values of the vertical segment. This tells us how many horizontal segments this vertical segment intersects. We simply add these counts together to get the desired global number of intersections. The running time is $O((n+m)\log(n+m))$.

First-Fit

You are packing up n items into boxes, and want to use as few boxes as possible. Each box can fit a total of 10 pounds of stuff, and the weight of the i^{th} item is $w_i \leq 10$. Your algorithm is this: initially, all the boxes are lined up, empty. You pick the next unpacked item (say item i), and put it in the *first* box that can hold the item (i.e., whose current weight is at most $10 - w_i$).

1. Argue that if OPT is the optimal number of boxes into which you can pack all the items, then your algorithm uses at most $2 \cdot OPT + 1$ bins.

   Solution: First, $OPT \geq \frac{1}{10}\sum_i w_i$, just by the weight restriction.

   Next, there cannot be two non-empty bins ending ≤ 5 pounds in our solution. Indeed, the items in the higher numbered bin could have been put into the lower numbered bins instead (they fit, because $5+5 \leq 10$). So at most one of our bins has ≤ 5, and the others are all more than 5. The latter number is $\leq 2OPT$ by the calculation above, and the former adds a 1.

2. How would you implement the algorithm in time $O(n \log n)$.

   Solution: Use a SegTree! From (a), we need at most $M = 2OPT + 1$ bins, so maintain a SegTree on M variables, the i^{th} variable indicating the *free space* in bin i. For each internal node of the SegTree maintain the *maximum* of the variables in its subtree. Now when item j comes, follow the left-most path in the SegTree consisting of numbers that are $\geq w_j$. And when the bin load changes, update the internal nodes appropriately.

VCG and Pricing Advertisements

We saw the VCG mechanism for incentive-compatible auctions in Lecture. Let's use this for pricing online advertising slots. There are 2 ad slots that ElGogo wants to sell on a page, the first slot has a clickthru rate of 0.5, the second has a clickthru rate of 0.3. *Each bidder can get at most one slot.* There are 4 bidders, with the following valuations:

- A: \$10 per click (so, e.g., this bidder values the first slot at 10*0.5 = 5, and the second slot at 10*0.3 = 3.)
- B: \$8 per click
- C: \$7 per click
- D: \$2 per click

1. What is the social-welfare maximizing allocation?

   Solution: A gets the first slot, B gets the second. The total value to A is 5 and the value to B is 8*0.3 = 2.4. Total social welfare = 7.4.

2. What are the VCG payments?

   Solution: If A did not bid, B and C would get the first and second slots respectively. The social welfare would be 8*0.5 + 7*0.3 = 6.1. So A's payment is how much his presence caused B,C,D's welfare to fall, = (optimal welfare without A) - (optimal welfare of everyone else with A) = 6.1 - (2.4 + 0 + 0) = 3.7.

   If B did not bid, A and C would get the first and second slots respectively. The social welfare would be 10*0.5 + 7*0.3 = 7.1. So B's payment is how much his presence caused the others' welfare to decrease = (optimal welfare without B) - (optimal welfare of everyone else with B) = 7.1 - (5 + 0 + 0) = 2.1.

   C and D do not pay anything.

Combinatorial Auctions

VCG can be used even with complicated preferences. Suppose we have two identical hotel rooms in Las Vegas, a flight ticket f from PIT to LAS, and a concert ticket c in Vegas to auction off. In the following, a generic hotel room is denoted by h, and none of the people want two rooms.

- Buyer A: values $\{h\}$ at \$100, $\{f\}$ at \$200, $\{h, f\}$ at \$450, $\{h, f, c\}$ at \$440. (He hates the band in question so much, he gets *negative value* from getting c along with h, f.) All other sets are valued at \$0.
- Buyer B (doesn't care for the concert): values $\{h\}$ at \$50, $\{f\}$ at \$400, $\{h, f\}$ at \$500, and $\{h, f, c\}$ at \$501. All other sets are valued at \$0.
- Buyer C (lives in Vegas): values $\{c\}$ (and all sets containing c) at \$200.

What is the social-welfare maximizing allocation, and what are the VCG payments?

Solution: The allocation is A gets $\{h\}$, B gets $\{h, f\}$, and C gets $\{c\}$. This gives a total valuation (aka social welfare) of $\$100 + 500 + 200 = \$800$.

A pays 0, B pays \$350, C pays \$1.